P.30

NASA Technical Memorandum 88331, Rev. 1

[redacted]

The Nature and Evaluation of Commercial Expert System Building Tools: Revision 1

William B. Gevarter

March 1987

National Aeronautics and
Space Administration

NASA Technical Memorandum 88331, Rev. 1

The Nature and Evaluation of Commercial Expert System Building Tools: Revision 1

William B. Gevarter, Ames Research Center, Moffett Field, California

March 1987

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

THE NATURE AND EVALUATION OF COMMERCIAL EXPERT SYSTEM BUILDING TOOLS: REVISION 1

William B. Gevarter

ABSTRACT

This memorandum reviews the factors that constitute an Expert System Building Tool (ESBT) and evaluates current tools in terms of these factors. Evaluation of these tools is based on their structure and their alternative forms of knowledge representation, inference mechanisms, and developer/end-user interfaces. Next, functional capabilities, such as diagnosis and design, are related to alternative forms of mechanization. The characteristics and capabilities of existing commercial tools are then reviewed in terms of these criteria.

INTRODUCTION

The development of new expert systems is changing rapidly both in ease of construction and time required due to improved Expert System Building Tools (ESBTs). These tools are the commercialized derivatives of artificial intelligence (AI) systems developed by AI researchers at universities and research organizations. It has been reported that these tools enable an order of magnitude less time to develop an expert system than would have been required with the use of traditional development languages such as LISP. This memorandum reviews the factors that make up an ESBT and evaluates current tools in terms of these factors. Because of the rapid changes in ESBTs, this memorandum has been written to revise and update an earlier version issued in June 1986.

THE STRUCTURE OF AN EXPERT SYSTEM BUILDING TOOL

The core of an expert system consists of a knowledge base and an accompanying inference engine that operates on the knowledge base to develop a desired solution or response. To use such a system, an interface is required to the end-user or to an array of sensors and effectors for communication with the relevant world. In addition, to facilitate the development of an expert system, an ESBT must also include a developer interface (1) so that the requisite knowledge base (KB) can be built for the particular domain application for which the system is intended; (2) to develop the appropriate end-user interface; and (3) to provide for any special instructions to the inference engine required for the particular domain. The character and quality of these interfaces function as one of the main differentiations between commercial tools and research ESBTs developed at universities. Also important in the structure of ESBTs are interfaces to

other software and data bases, and the computers on which they will run for development and for delivery. Figure 1 summarizes the structure of an ESBT.

KNOWLEDGE REPRESENTATION

The type of knowledge that can be easily represented by the tool is a key consideration in choosing ESBTs. As indicated by Figure 2, there are three aspects of Knowledge Representation (KR) that are fundamental in these tools — object descriptions (declarative knowledge, facts), certainties, and actions. One method of representing objects is by frames (with or without "inheritance"[1]). Frames are data structures for representing stereotyped objects or situations. A frame has slots to be filled with data for objects and relations appropriate to the situation. A version of frames incorporating provisions for message-passing between objects includes procedures that can be activated by the received messages, and therefore supports "object-oriented programming," is referred to in the figure as objects. Declarative knowledge can also be represented by parameter value pairs, by use of logic notation, and to some extent by rules.

"Actions" change a situation and/or modify the relevant data base. Actions are most commonly represented by rules. These rules may be grouped together in modules, in terms of applicability, for easy maintenance and rapid access. Actions may also be represented in terms of examples that indicate the conclusions or decisions reached. Examples are a particularly desirable form of representation for facilitating knowledge acquisition and are capitalized upon in inductive systems. Examples are much easier to elicit from experts than rules, and may often be a natural form of domain knowledge. Actions can also be expressed in logical notation —a form of rule representation. Finally, actions can be expressed as procedures elicited by messages in object-oriented programming or by changes in a global data base observed by "demons" (procedures that monitor a situation and interrupt to perform an action when their activating conditions appear).

In addition to the representation of objects and actions, one must consider the degree to which the knowledge is known to be correct. Thus, most ESBTs have provisions for representing certainty. The most common approach is to incorporate "confidence factors," a derivative of the approach used by the MYCIN expert system (Shortliffe, 1976); "fuzzy logic" and probability are also used. An alternate way of handling uncertainties is to consider multiple worlds in which different items are true or not true. Another consideration is whether or not a "deep model" of the system can readily be built with the tool to enable model-based reasoning. (The same underlying model can often be employed for multiple purposes.) Finally, system size (e.g., number of rules needed) can be of critical importance as it can have an important effect both on memory requirements and memory management, and on run times.

INFERENCE ENGINE

Figure 3 indicates the major alternative means of doing inferencing that might appear in an ESBT. The most usual approach is backward chaining, where given a fixed number of possible

[1]Inheritance allows knowledge bases to be organized as hierarchical collections of frames which inherit information from frames above them in the hierarchy. Thus inheritance mechanisms provide a form of inference.

conclusions, hypothesized conclusions are evaluated to see whether the evidence supports them. This evaluation is usually done by backward chaining through the rules; starting with rules that have the hypothesized conclusions as their outcome (consequent). Rules are then searched for those that have as their outcome conditions that support the input conditions (antecedents) in the hypothesized conclusion rule. This process continues recursively until the hypothesis is fully supported, or until a negation or dead end is reached. If the latter happens, additional hypotheses may be tried until some conclusion is reached or the process is terminated. This depth-first, backward-chaining approach was popularized by the MYCIN expert system. The corresponding EMYCIN ESBT shell (Van Melle, 1980) is the prototype of virtually all the hypothesis- (or goal-) driven commercial ESBTs currently available.

Forward chaining starts with input data or with the situation currently present in a global data base. The data or the situation is then matched with the input conditions in each of the relevant rules to determine applicability for the current situation (usually represented by a set of attributes and their associated values). One of the matching rules is then selected (e.g., by the use of meta-rules or priorities), and the rule's consequents are used to add information to the data base or to actuate some procedure that changes the global situation. Forward chaining also proceeds recursively (similar to backward chaining), terminating when a desired result or conclusion is reached, or when all relevant rules are exhausted. Combinations of forward and backward chaining have also been found useful in certain situations.

Forward reasoning (a more general form of forward chaining) can be done with data-driven rules or data-driven procedures (demons).

Hypothetical reasoning refers to solution approaches in which assumptions may have to be made to enable the search procedure to proceed. However, later along the search path, it may be found that certain assumptions are invalid and therefore have to be retracted. This "nonmonotonic" reasoning (reasoning in which facts or conclusions must be retracted given new information) can be handled in a variety of ways. One approach that reduces the computation required is to carry along multiple solutions (representing different hypotheses) in parallel and to discard inappropriate ones as contradictory evidence is gathered. This approach is referred to as "viewpoints," "contexts," or "worlds" in different tools. Another approach is to keep track of the assumptions that support the current search path, and backtrack to the appropriate branch point when the current path is invalidated. This latter approach has been referred to by such names as "nonchronological backtracking." A related capability is "truth maintenance," which removes derived beliefs when their conditions are no longer valid.

Object-oriented programming is an approach in which information about an object and procedures appropriate to that object are grouped together into a data structure such as a frame. These procedures are actuated by messages which are sent to the object from a central controller or another object. This approach is particularly useful for simulations involving a group of distinct objects, and for real-time signal processing.

The blackboard inference approach is associated with a group of cooperating expert systems that communicate via shared information on a common data structure referred to as a blackboard. An agenda mechanism can be used to facilitate the control of the solution development on the blackboard.

In ESBTs, logic commonly refers to a theorem-proving approach involving "unification." Unification refers to substitutions for variables that make two items identically matched. The common logic implementations are versions of Prolog that utilize a relatively exhaustive depth-

first search approach.

An important inference approach found in some tools is the ability to inductively generate rules or "decision trees" from examples. As human experts are often able to better articulate their expertise in the form of examples rather than in the form of rules, inductive learning techniques are frequently ideal methods of knowledge acquisition for rapid prototyping. The resultant expert system can then be iteratively refined by their human builders by critiquing and modifying the results produced by the system. Inductive inference usually proceeds by starting with one of the input parameters and searching for a tree featuring the minimum number of decisions needed to reach a conclusion. The minimum-depth tree is found by cycling through all parameters as possible initial nodes and using an "information theoretic" approach to selecting the order and necessity of the parameters to be used for the remaining nodes. The depth of the tree is usually relatively shallow (often a depth less than five) so that large numbers of examples usually result in broad, shallow trees.

Some tools incorporate demons which monitor local values and execute procedures when their actuation conditions appear. These are particularly appropriate for monitoring applications.

A number of tools offer a choice of several possible inference or search procedures. In such systems, means are usually made available to the system builder to control the choice of the inference strategies, depending on the system state. Such control is referred to as meta-control. One form of meta-control is the use of "control blocks," which are procedures that tell the system generically the next steps to take in a given situation, so that the search is reduced, and a large number of rules can be accommodated without the search space becoming combinatorially explosive.

As the certainty of data, rules, and procedures is usually less than 100%, most systems incorporate facilities for certainty management. Thus, they have various approaches for combining uncertain rules and information to obtain a certainty for the result.

Pattern matching is often required to mechanize inference techniques. The sophistication of the pattern matching approaches affects the capabilities of the system. Pattern matching varies from matched identical strings to variables, literals, and wildcards, and even partial and approximate matching that serves as analogical reasoning.

Other ESBT capabilities vary from tool to tool. Some inference engines offer rapid and sophisticated math calculation capabilities. One of the more valuable capabilities is supplied by inference engines that can manage modularized knowledge bases or solution subproblems by accessing and linking these as needed.

Another important consideration in a tool is the degree of integration of its various features. Full integration is desirable so that all the tool features can be brought to bear, if needed, in the solution of a single problem. For example, it is desirable, when appropriate, that expert system developers be able to freely mix forward and backward chaining rules and be able to reason about information stored in objects, for ESBTs having these features.

DEVELOPER INTERFACE

Various tools offer different levels of capabilities by which the expert system builder can mold the system. The simpler tools are shells into which knowledge is inserted in a specific structured fashion. The more sophisticated tools are generally more difficult to learn, but

allow a much wider choice for the system developer. In these, the developer can choose among various knowledge base representations, inference strategies, and the form of the end-user (or system) interface. Various levels of debugging assistance are also provided. Figure 4 provides an indication (dependent on the tool) of the possible options that are available for each aspect of the developer's interface.

END-USER INTERFACE

Once the expert system has been built, its usability will depend in large part on the end-user interface. Figure 5 indicates some of the range of end-user facilities found in ESBTs. As most expert systems are really intelligent assistants, the end-user interface is often designed as an interactive dialogue. This dialogue and/or initial input most often appears to the user as a structured data-input arrangement that incorporates menu choices to answer requests by the system for information. In some cases, systems will accept multiple and uncertain user responses and still arrive at conclusions (though with reduced certainty), which increases flexibility. For sophisticated systems, graphics are often used to show the line of reasoning in response to users' "how" questions; whereas in simpler systems a listing of the rules supporting the system's conclusion may be employed. The "Why do you need this information?" questions by a user in response to queries by the system are often justified by quoting the rule for which the information is required. The ability of the system to answer the user's "why" and "how" questions is important, for it increases the end-user's confidence in the system's decision-making ability.

Other aspects often found in ESBTs are facilities that allow the end-user to change parameters ("what if" queries) and observe the affect on the outcome; facilities to allow the user to initially prune the line of questioning or search so that the system need not pursue areas that the user feels are irrelevant or unnecessary, and the capability to save examples for future consideration or use.

Very sophisticated tools will often include interactive graphics and simulation facilities to increase the end user's understanding and control of the system being represented. It is very important that the end user interface be "user-friendly" to facilitate system acceptance.

ASPECTS OF TOOL SOFTWARE AND COMPUTERS SUPPORTED

In addition to the structure and the paradigms supported by a tool, the language in which the tool is written is of major importance. The language determines whether the expert system is compilable and, if so, incrementally or in a batch mode. Compilability reduces memory requirements and increases the speed of the expert system; incremental compilability speeds development. Figure 6 is illustrative of the aspects related to the tool language choice.

In general, the more sophisticated tools have been written in LISP. However, even these tools are now being rewritten in languages such as C to increase speed, reduce memory requirements, and to increase availability on a larger variety of computers. However, some new approaches to mechanizing LISP may reduce the speed and memory advantages associated with C.

Tools written in LISP are generally extensible by the user by writing additional LISP functions. This is also true of some of the other languages, e.g., Prolog and PASCAL. Similar

considerations relate to the tool having language hooks for accessing other programs, or data base hooks for accessing other information. In some cases the expert system generated by the tool is fully embeddable in other systems for more autonomous operations. This latter consideration is becoming increasingly important now that expert systems are moving from prototypes to being fielded. Reliability and memory management (e.g., for LISP garbage collection) are often important considerations for fielded systems.

The computers supported by the various tools are primarily a function of the language and operating system in which they are written, and the memory, processing and graphic display capabilities of the individual computers. The trend toward making expert system shells available on personal computers (such as the IBM) is partially due to the increasing capabilities of these computers. However, this trend is also in part due to writing tools in faster languages, such as C, and to taking advantage of modularization in building the knowledge base — decomposing the problem into subproblems, and providing appropriate linking, as required, during operation.

FUNCTIONAL CAPABILITIES

Of primary consideration are the functional applications that can readily be built with a particular ESBT. A review of the major functional applications (outlined in Figure 7) follows:

1. CLASSIFICATION

   By far the most common function addressed by expert systems is classification. Classification refers to selecting an answer from a fixed set of alternatives, based upon input information. Subcategories of classification include:

   - Interpretation of Measurements: This refers to hypothesis selection, given measurement data and corollary information.
   - Diagnosis: In diagnosis, the system not only interprets data but seeks additional data, as required to aid its line of reasoning.
   - Debugging, Treatment, or Repair: These functions refer to taking actions or recommending measures to correct an adverse situation that has been diagnosed.
   - Use Advisor: An expert system as a front end to a computer program or to a piece of machinery can be very helpful to the inexperienced user. Such systems depend on the goals of the user and the current situation in suggesting what to do next. Thus, the advice evolves as the state of the world changes. Use advisors can also be helpful in guiding users through procedures in other domains (e.g., auto repair and piloting aircraft).

   Classification (as well as other functional applications) can be considered to be of two types: surface reasoning and deep reasoning. In surface reasoning no model of the system is employed, the approach taken is to write a collection of rules asserting that a certain situation warrants a certain response or conclusion (usually written as a heuristic rule garnered from experience). In deep reasoning, the system draws upon models or structures of the system to help arrive at the conclusion. Thus, systems employing deep models are potentially more capable and may degrade more gracefully than those relying on surface reasoning.

2. DESIGN AND SYNTHESIS

Design and synthesis refers to configuring a system based on a set of alternative possibilities. The expert system incorporates constraints that the system must meet as well as guidance for steps to meet the user's objectives.

3. INTELLIGENT ASSISTANT

Here the emphasis is on having a system that can give advice, furnish information, or perform various subtasks, depending on user needs.

4. PREDICTION

Prediction refers to forecasting what will happen in the future based on current information. This forecasting may depend upon experience alone or it may involve use of models and formulas. The more dynamic systems may use simulation to aid in the forecasting.

5. SCHEDULING

Scheduling refers to time-ordering a given set of tasks so they can be done within the resources available, without interfering with each other.

6. PLANNING

Planning is the selection of a series of actions from a complex set of alternatives to meet the user's goals. It is more complex than scheduling in that tasks are chosen, not given. In many cases, time and resource constraints do not permit all goals to be met. In this case the most desirable outcome is then sought.

7. MONITORING

Monitoring refers to observing an ongoing situation for its progress as predicted or intended, and alerting the user or system if there is a departure from the expected or usual. Typical examples may be space flights, industrial processes, patients' conditions, and enemy actions.

8. CONTROL

Control is a combination of monitoring a system, and taking appropriate actions in response, to achieve desired goals. In many cases, such as the operation of vehicles or machines, the tolerable response delay may be as small as milliseconds. Thus, the system may be referred to as a "real-time" system. Real-time is defined as responding within the permissible delay time, so that the system being controlled stays within its operating boundaries.

9. DIGEST OF INFORMATION

Such a system may take in information and return a new organization or synthesis. One application may be the determination of a decision tree from examples. Another may be situation assessment of a military situation or the stock market, based on input data and corollary information.

10. DISCOVERY

    Discovery is similar to Digest of Information except that the emphasis is on finding new relations, order, or concepts. This is still a research area. Examples include finding new mathematical concepts and elementary laws of physics.

11. OTHERS

    There are other functions, such as learning, that are directly subsumable under the ones we have enumerated thus far. In many cases these functions (and some of those already mentioned) can be ingeniously decomposed into ones discussed previously. Thus, for example, design and some other functions can often be separated into subtasks that can be solved by classification.

IMPORTANCE OF VARIOUS ESBT ATTRIBUTES FOR PARTICULAR FUNCTIONAL APPLICATIONS

Table 1 is an attempt to relate the various attributes that are found in different ESBTs to their importance in facilitating the building of expert systems that perform different functions.[2] A solid oval indicates an attribute that is perceived as very worthwhile in helping to build that function. An open oval indicates that it is a lesser contributor. An empty cell indicates an attribute that does not provide a significant contribution. As indicated earlier, the evaluation is subjective, as some of the functions can be decomposed into other functions dependent upon the insights and ingenuity of the system developer. Thus, Table 1 reflects what the author sees as obvious (and perhaps necessary) attributes for straightforward construction of expert systems that perform the indicated functions.

ATTRIBUTES OF PARTICULAR COMMERCIAL ESBTs

Appendix A presents brief descriptions of some of the better-known commercial ESBTs. Attributes of these ESBTs are listed in Appendix B. Inclusion of an ESBT in this memorandum in no way represents an endorsement of that product. The descriptions and listing have been constructed from company literature, discussions with company representatives, open-literature sources, demonstrations, exploratory use of the tools, etc. However, some incompleteness, errors, and oversights are inevitable in such an endeavor, so it behooves the potential purchaser to use this memorandum as a guide and to examine directly those systems in which he or she is interested. Direct examination is particularly important because increasing competition is forcing ESBT developers to make rapid improvements and changes in both their systems and their prices.

[2]Various approaches to ESBTs may be shown to be equivalent to a Turing Machine so that any computation can be performed by them. Therefore, it usually cannot be said definitively that ESBT x cannot do function y. Thus, Table 1 is really an attempt to reflect the author's perception of which ESBT attributes facilitate or simplify the programming of various expert system functions.

TABLE 1.– A SUBJECTIVE VIEW OF THE IMPORTANCE OF VARIOUS EXPERT SYSTEM TOOL ATTRIBUTES FOR PARTICULAR FUNCTIONAL APPLICATIONS

● VERY ○ SOMEWHAT LITTLE	DIAGNOSIS AND CLASSIFICATION	DATA ANALYSIS AND INTERPRETATION	DESIGN AND SYNTHESIS	PREDICTION AND SIMULATION	MONITORING	USE ADVISOR	INTELLIGENT ASSIST.	PLANNING AND SCHEDULING	CONTROL
INFERENCE APPROACH									
BC	●	○	○		○	○	○	○	○
FC & FORWARD REASONING	○	●	●	●	●	●	○	●	●
BC, FC, & FORWARD REASON.	●	●	●	○	○	○	●	●	○
HYPOTHETICAL REASONING	●	●	●	○	○	○	○	●	
OBJECT-ORIENTED	○	○	○	●	○	○	○	○	●
BLACKBOARD	●	●	●	○	○	○	○	●	
INDUCTION	●	○	○			○	○		○
OBJECT DESCRIPTION									
FRAMES	●	○	●	○	○	○	○	●	○
FRAMES W/ INHERITANCE	●	○	●	○	○	○	○	●	○
OBJECTS	○	○	○	●	○	○	○	○	●
PARAMETER VALUES PAIRS	○	○	○		○	○			○
LOGIC	○	○	○	○	○	○	○	○	○
RULES	○	○	○	○	○	○	○	○	○
CERTAINTIES	●	●	○	○	○	○	○	○	○
ACTIONS									
RULES	●	●	●	●	●	●	●	●	●
EXAMPLES	●	○	○	○		○	○		○
LOGIC	●	●	●	●	○	○	○	○	○
MESSAGES	○	○	●	●	○	○	○	●	●
PROCEDURES	○	○	●	●	●	○		●	●

COMPARATIVE COMPOSITE VIEW OF THE VARIOUS TOOLS

Table 2 provides a composite view of the various ESBTs. Many of the attributes have been integrated to provide a more easily understandable picture of the capability of the tools in each subcategory (e.g., representation of actions and ease of knowledge base creation). A solid oval indicates that the tool appears strong in a subcategory, an open oval indicates fair, and an empty cell indicates little or no capability in that area. Note that by relating each tool's attributes to its functional importance, an attempt has been made to indicate each tool's suitability for developing various functional applications. Also, note that the more expensive and correspondingly more sophisticated tools have the widest applicability. This is often due to their being a collection of different paradigms incorporated into a single tool. As a result, they may often be regarded as a higher-order programming language and environment, instead of as a simple shell into which information is inserted, and an expert system directly results. The latter is more nearly true of the simpler induction systems which can be considered as knowledge acquisition and rapid prototyping tools from which more complex systems can be built using other tools and the rules thus far generated.

OVERALL USABILITY OF A TOOL

Figure 8 summarizes some of the aspects that enter into the critical ESBT attribute "overall usability of a particular tool." In addition to obvious factors, such as costs and functional applicability (which functions are easily accomplished with the tool and which are difficult) , tool choices should be guided by size of the system to be built, how rapidly a system of the given size and complexity can be built with the tool, and the speed of operation of the tool during development and particularly during end-use (when functioning as a delivery vehicle[3] for the developed expert system). Perhaps the most important factor, however, is user satisfaction both of the developer and the end user. This is related to how obvious are the various functions to use, how direct is the line of action to the user's goals, the control the user senses that he/she has over the system, the nature of the interaction or display (e.g., via menu or graphics), how easy is it to recover from errors, the on-line help that is furnished, and the perceived esthetics, reasonableness, and transparency of the system. Also of major importance is how easy it is to learn the system. This often depends on many of the above factors already discussed, but is also closely related to how apparent the choice is at each step (e.g., use of menus versus via programming), the quality of the documentation and on-line help, and the system's structure. Manufacturer-sponsored courses help; however, these are often expensive and inconvenient. A related factor is manufacturer support of the tool, particularly the availability of help via the telephone when required.

Finally, such factors as the system's portability, machines it will run on, the delivery environment, its capability to interface with other programs and data bases, and whether the developed system can be readily imbedded in a larger system will all be important in an evaluation of a tool. A more difficult factor to evaluate is the ease of prototyping versus life cycle cost. As

[3]In some cases, the more sophisticated ESBTs are used only for development of an Expert System, with the final end user version being reprogrammed in a more portable language such as C.

TABLE 2.– COMPOSITE VIEW OF COMMERCIAL EXPERT SYSTEM BUILDING TOOLS

● STRONG
○ FAIR
□ USER PROG.

SYSTEM	KNOW. REP.				INFERENCE ENGINE							FUNCTIONAL CAPABIL.					DEV. INTERFACE				END USER INTER.				S/W H/W CHARAC.				
	OBJ. REP.	CERTAINTIES	ACTIONS	EXAMPLES	PATTERN MATCH.	BC	FC	OBJ.-ORIENTED	LOGIC	INDUCTION	HYPO. REASON.	CLASS. INTERPRETA. ADVICE (SHALLOW)	CLASS. INTERPRETA. ADVICE (DEEP)	DESIGN	PLANNING	MONITOR/CONTROL	KB CREATION	DEBUGGING	GRAPHICS	USER INT. CRTL.	INFER. CONTROL	WHY/HOW	SCREEN INTER.	GRAPHICS	WHAT IF	TOOL LANG. **	@ COMPILABLE	LANG/OS HOOKS	COMPUTERS @@	1st UNIT COST $K
ART	●	□	●		●	●	●	●	●		●	●	●	●	●	○	●	●	●	●	○	●	●	●	●	L	D, IN	○	V, Su, S, L, T	65
KEE	●	□	●		●	●	○	●	●		●	●	●	●	●	○	●	●	●	●	●	●	●	●	●	L	IN	○	A, S, L, T, X, O	52† + SIM KIT
KNOWLEDGE CRAFT	●	□	●		●	●	●	●	●		●	●	●	●	●	○	●	●	●	●	□	●	●	●	○	L	IN	○	L, S, T, V, O	50† + SIM
PICON	●		●		●	●	●	●				●	●			●	●	●	●	●	○	●	●	●	●	L, C	Y	●	L, T	60
S. 1	○	C	●		○	●	FR		○			●	●	○			●	●	●	●	●	●	●	○	●	C	Y	●	370, V, Su, H, O	25, 45
ES ENVIRON/VM OR VMS		C, F	●		○	●	○					●				○	●	○		●	○	●	○		●	PA	IN	●	370	60
ENVISAGE (ADVISOR)	○	F, B	●		○	●	○					●	○				○	●		○	○	●	○		○	PA	Y	●	V, μV	25, 15
KES		C, B	●		○	●	○				●	●	○				○	○		○		●	○		●	C	Y	○	I, A, V, O, Su, μV	4, 7†, 25†
M. 1		C	●		○	●	FR		○			●	○	○	○	○	●	●		●	○	●	○			C	Y	●	I	5
NEXPERT OBJECT	●	□	●		○	●	●					●	○	○	○	○	●	●	●	●	●	●	●	●	●	C	IN	●	I, M, O	3, 5
PERSONAL CONSULTANT +	○	C	●			●	○					●	○				●	●	○	●	●	●	●	○	●	L	Y	●	I, TP, T	3
EXSYS 3.0		C	○		○	●	○					○					●	●		○		●	●	○	●	C	IN	○	I, V	0.4, 5
EXPERT EDGE		B	○		○	●						●					●	●		●		●	●		○	C	Y	○	I	2.5, 5
ESP FRAME ENGINE	●		●		○	●	○		●			●	●				●	●	○	○	○	●	●			PR	IN	○	I	0.9
INSIGHT 2 +		C	○		○	●	○					●					●	●		○		●	○		●	PA	Y	●	I, O	0.5
TIMM		C, I	○	●	●		●			●		●	○			○	●	●		○		○	●	●	○	F	R	●	TP, I, V, O	1.9, 19
RULEMASTER 3.0		F	○	●	○	●	○			●		●		○		○	●	●		○		○	○			C	DT	●	I, A, Su, V	1, 5, 17.5
KDS 3	○	C		●		●	●	○		●		●	○	○	○	○	●	○	○	●	○	○	●	●	○	A	R	●	I, TP	1.5
1st CLASS		C	○	●	○	○	○			●		●					●	●	○	○		○	○		○	PA	DT	●	I, TP	0.5
OPS5+			●		●		●					●	○	○	○	○	●	●			○		○			C	Y	○	A, Su, I, M, O	1.8, 3

FR — FORWARD REASONING

* C — CONFID. FACTOR
F — FUZZY LOGIC
B — BAYSIAN LOGIC
I — INCOMPLETE INFO

** L — LISP
PA — PASCAL
A — ASSEMBLY
F — FORTRAN
PR — PROLOG
O — OTHERS
C — C

@ D — DISCRIM. NET
R — RULES
IN — INCREMENTAL
Y — YES
DT — DECIS. TREE

@@ 370 — IBM 370
M — MACINTOSH
TP — TI PC
I — IBM PCs
S — SYMBOLICS
L — LMI
T — TI EXPLORER
X — XEROX
V — VAX
μV — MICROVAX
Su — SUN
A — APOLLO
O — OTHERS

† — INCLUDES TRAINING

prototypes are expanded into fielded systems, and iteratively further expanded and updated, difficulties are often encountered in system stability, run time, and memory management.

Though many of these factors can be deduced from the tool's specifications and system demonstrations, in many cases two tools intended for the same applications can be properly differentiated only by learning both systems and attempting to build the same set of applications with the two systems. (Appendix C provides an indication of this approach with respect to deriving timing benchmarks for ART, KEE, and several versions of OPS5 for a particular example, the monkey and bananas problem.) Nevertheless, the factors described in this memorandum, and the initial evaluation furnished, should prove useful in initially guiding potential tool users.

BIBLIOGRAPHY

Bundy, A., Ed.; *Catalogue of Artificial Intelligence Tools*, Springer-Verlag, Springer-Verlag, New York, 1985.

Gevarter, W. B.; *Intelligent Machines*, Prentice Hall, Englewood Cliffs, NJ, 1985.

Gilmore, J. F. and Pulaski, K.; A Survey of Expert System Tools, *Proc. The Second Conference on Artificial Intelligence Applications*, Miami Beach, FL, Dec. 11-13, 1985, pp. 498-502.

Gilmore, J. F., Pulaski, K., and Howard, C.; A Comprehensive Evaluation of Expert System Tools, *Proc. SPIE Applications of Artificial Intelligence*, Orlando, FL, April 1986.

Harmon, P. and King, C. D.; *Artificial Intelligence in Business*, John Wiley, New York, 1985.

Hayes-Roth, F; Waterman, D. A. and Lenat, D. B; *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983.

Karna, Animesh and Karna, Amitabh; Evaluating Existing Tools for Developing Expert Systems in PC Environment, *Proc. Expert Systems in Government*, Karna, K. N., Ed., McLean, VA, Oct. 24-25, 1985, pp. 295-300.

Riley, G. D.; Timing Tests of Expert System Building Tools, NASA JSC Memorandum, FM7/Artificial Intelligence Section, April 3, 1986.

Richer, M. H.; An Evaluation of Expert System Development Tools, *Expert Systems*, Vol. 3, No. 3, July 1986.

Shortliffe, E. H.; *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York, 1976.

Van Melle, W.; *A Domain Independent System that Aids in Constructing Knowledge-Based Consultation Programs*, Report No. 820, Computer Science Dept., Stanford University, 1980.

Waterman, D. A.; *A Guide to Expert Systems*, Addison-Wesley, Reading, MA, 1986, pp. 336-379.

Small Expert Systems Building Tools, *Expert Systems Strategies*, Harmon, P., Ed., Vol. 1, No. 1, Sept. 1985, Cahners Publishing Co., Newton, MA, pp. 1-10.

Expert Systems-Building Tools, *Expert Systems Strategies*, Harmon, P., Ed., Vol. 2, No. 8, Aug. 1986, Cutter Information Corp., Arlington, MA, pp. 17-24.

AI Development on the PC: A Review of Expert System Tools, *The Spang Robinson Report*, Vol. 1, No. 1, Nov. 1985, pp. 7-14.

Expert Systems Issue, *PC*, Vol. 4, No. 8, Apr. 16, 1985, pp. 108-189.

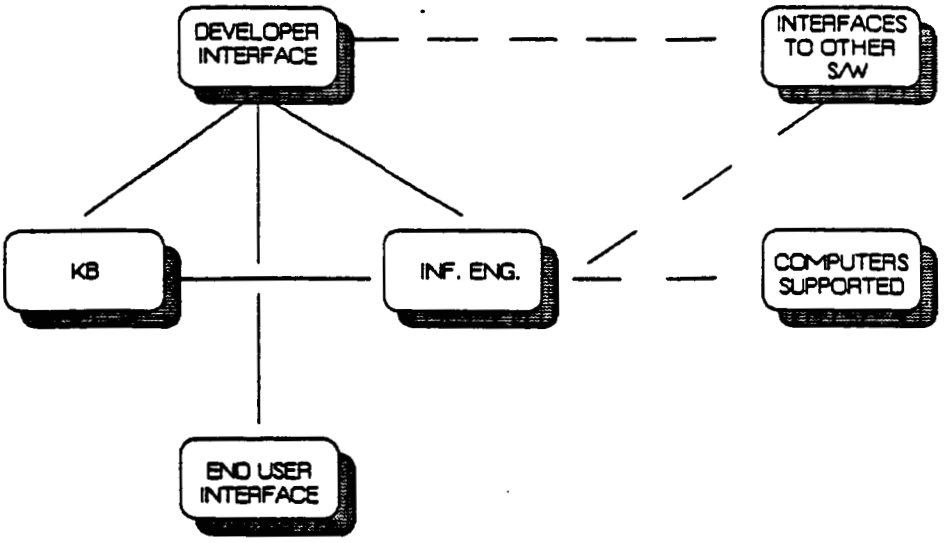

Figure 1.- An Expert System Building Tool structure.

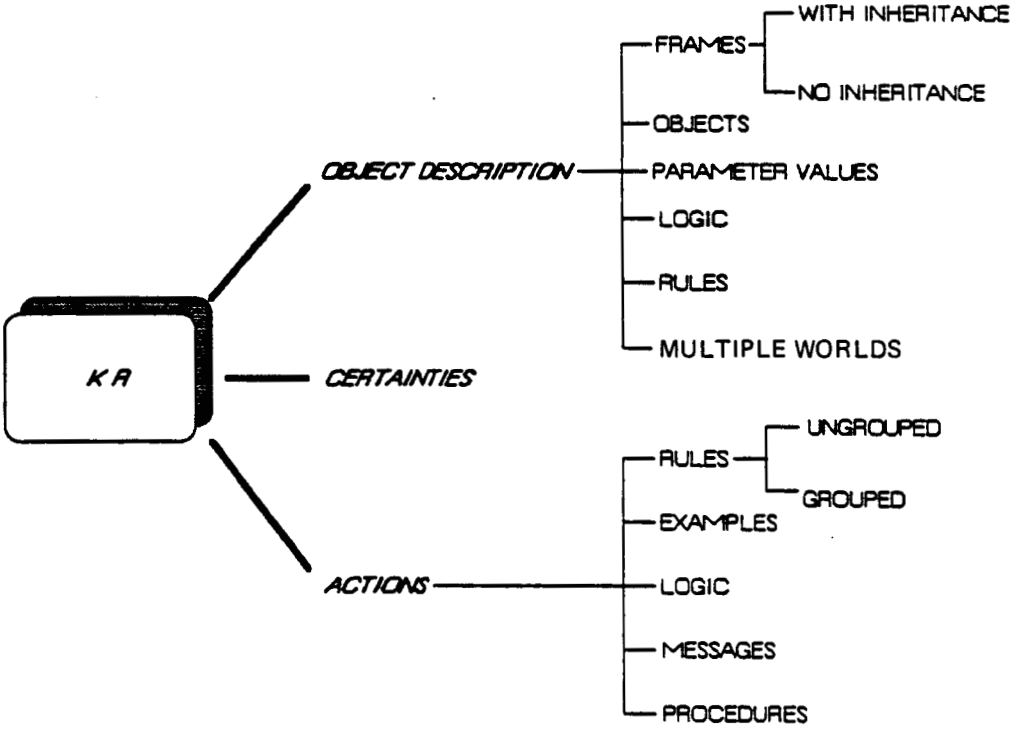

Figure 2.- Knowledge representation possibilities.

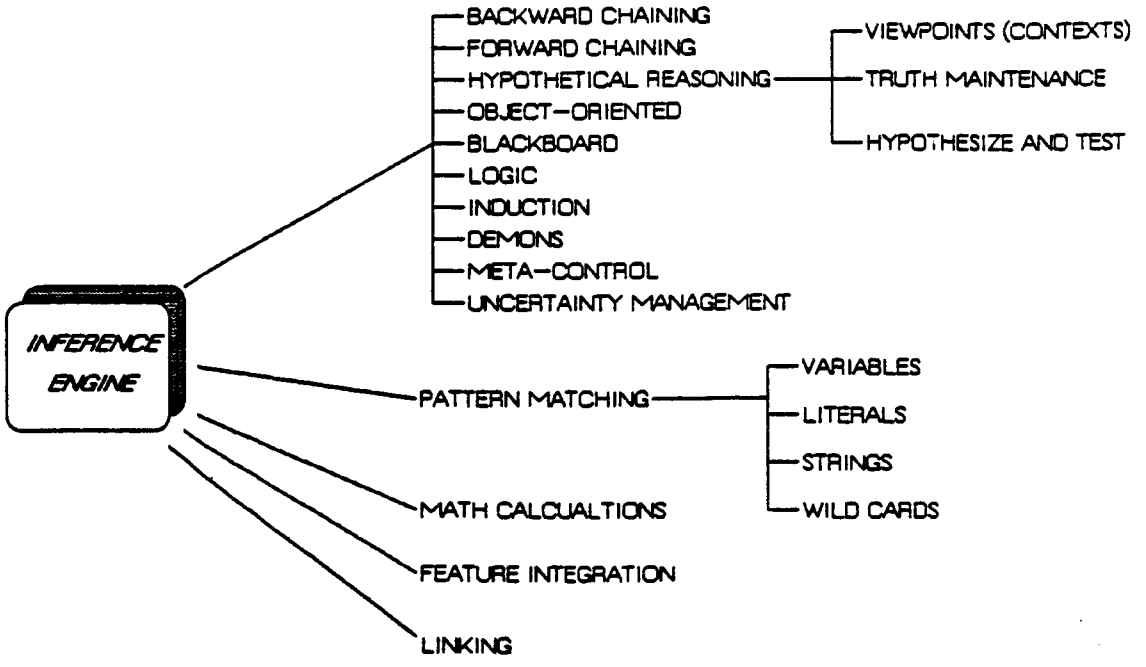

Figure 3.- Inference engine possibilities.

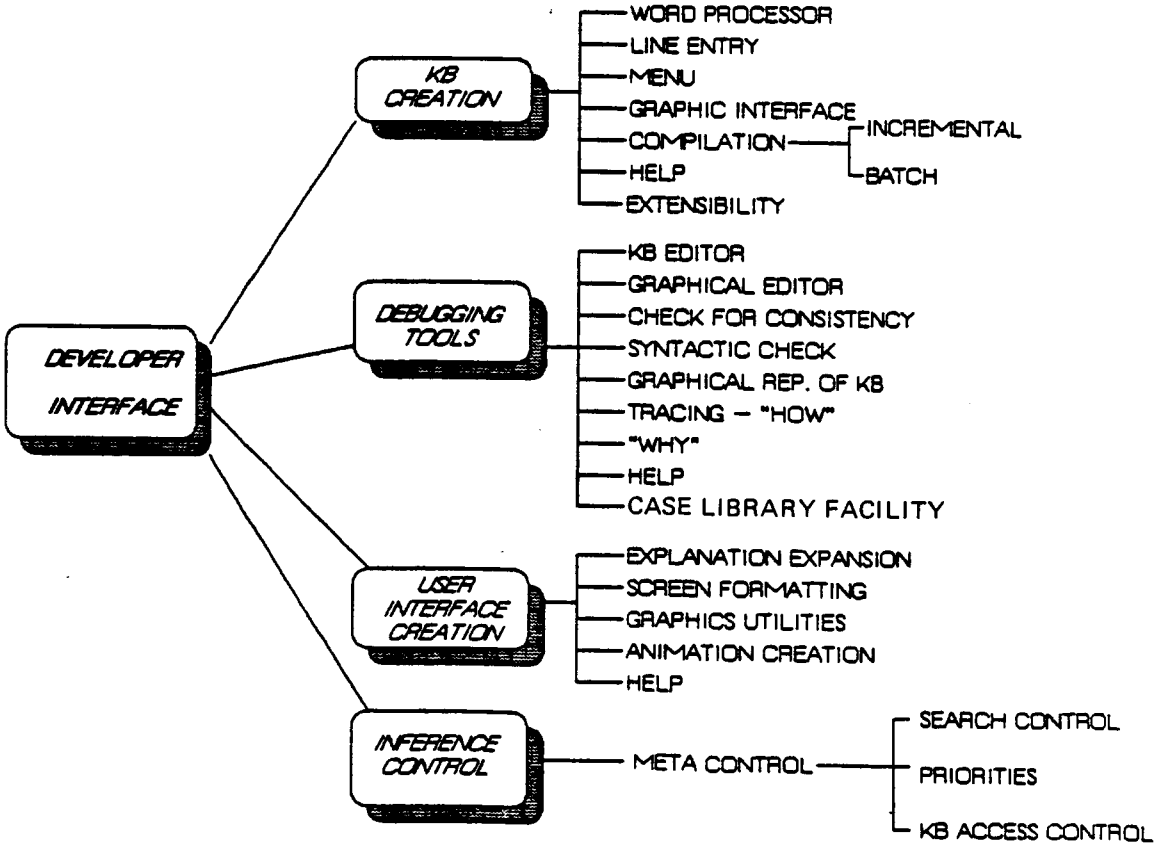

Figure 4.- Developer interface possibilities.

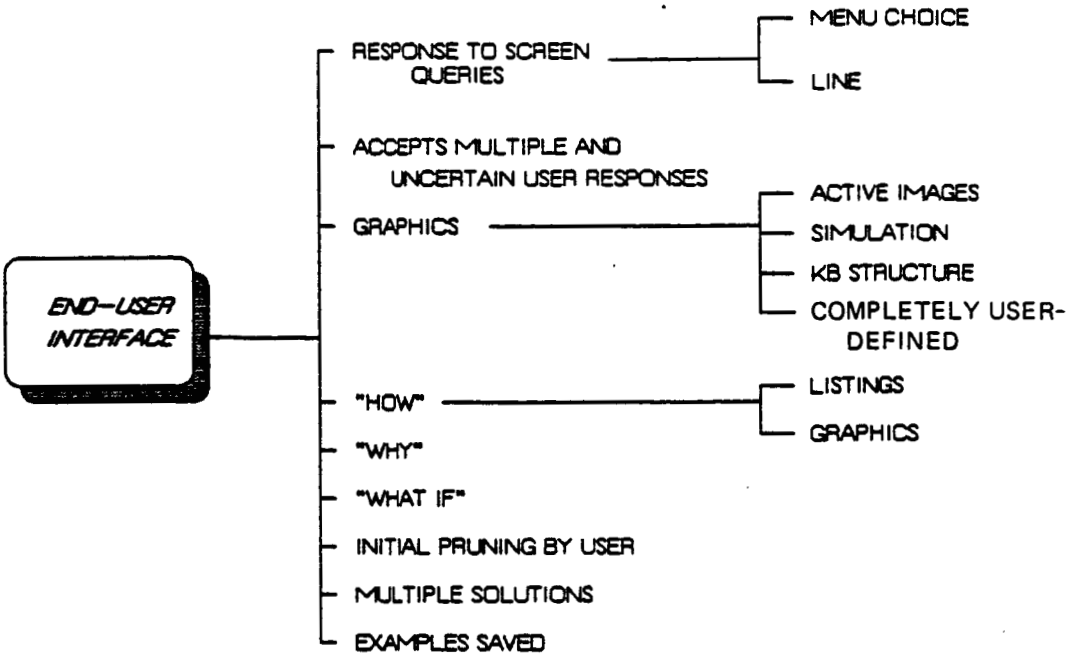

Figure 5.- End-user interface possibilities.

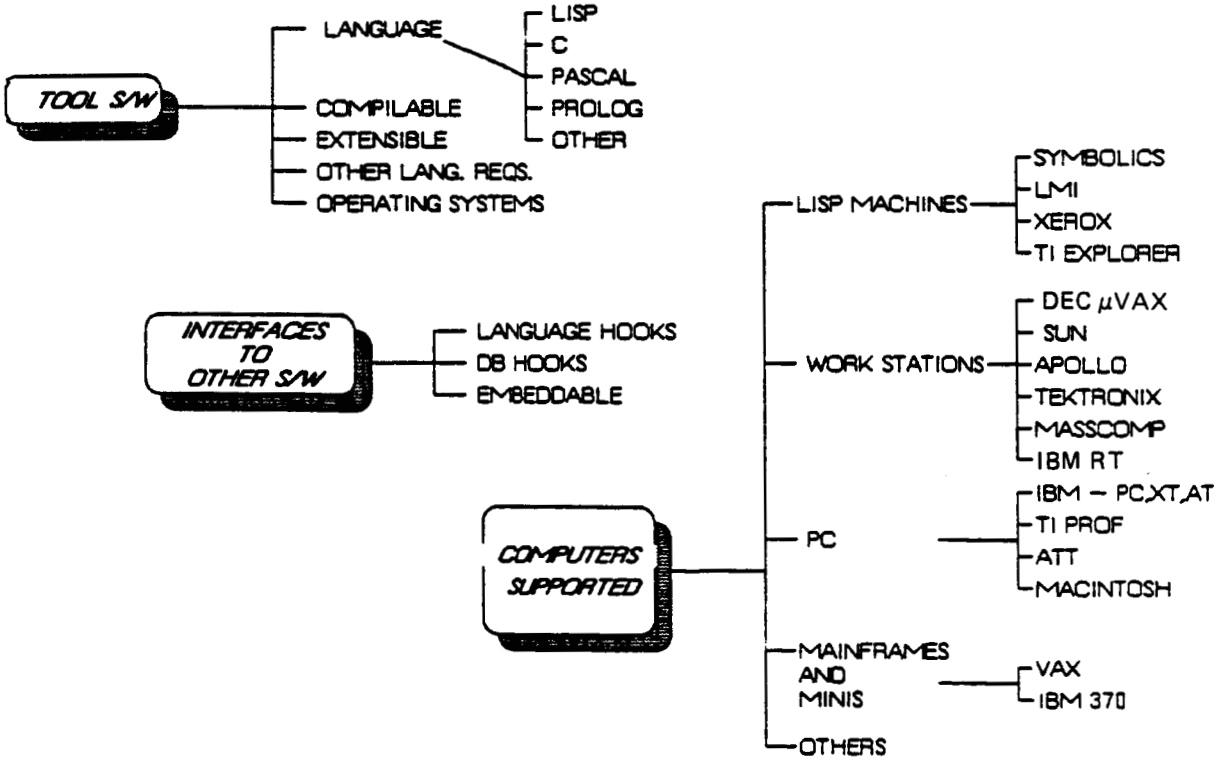

Figure 6.- Tool software and hardware aspects.

1. CLASSIFICATION
   - *INTERPRETATION OF MEASUREMENTS*
     HYPOTHESIS SELECTION BASED ON EVIDENCE
   - *DIAGNOSIS*
     MEASUREMENT SELECTION, INTERPRETATION
     - OFTEN USES MODEL OF SYS. ORG. AND BEHAVIOR
2. DESIGN AND SYNTHESIS
   PROVIDE CONSTRAINTS AS WELL AS GUIDANCE
3. PREDICTION
   FORECASTING
4. USE ADVISOR
   HOW TO
5. INTELLIGENT ASSISTANT
   DECISION AIDS
6. SCHEDULING
   TIME-ORDERING OF TASKS
   WITHIN RESOURCE CONSTRAINTS
7. PLANNING
   MANY COMPLEX CHOICES AFFECT
   EACH OTHER
8. MONITORING
   REAL-TIME, RELIABLE OPERATION
9. CONTROL
   PROCESS CONTROL
10. INFORMATION DIGEST
    SITUATION ASSESSMENT
11. DISCOVERY
    NEW RELATIONS OR CONCEPTS
12. DEBUGGING
    CORRECTING ACTION
13. EXAMPLE-BASED REASONING
    SOURCE OF GENERALITY OF RULES

Figure 7.- AI functional capabilities.

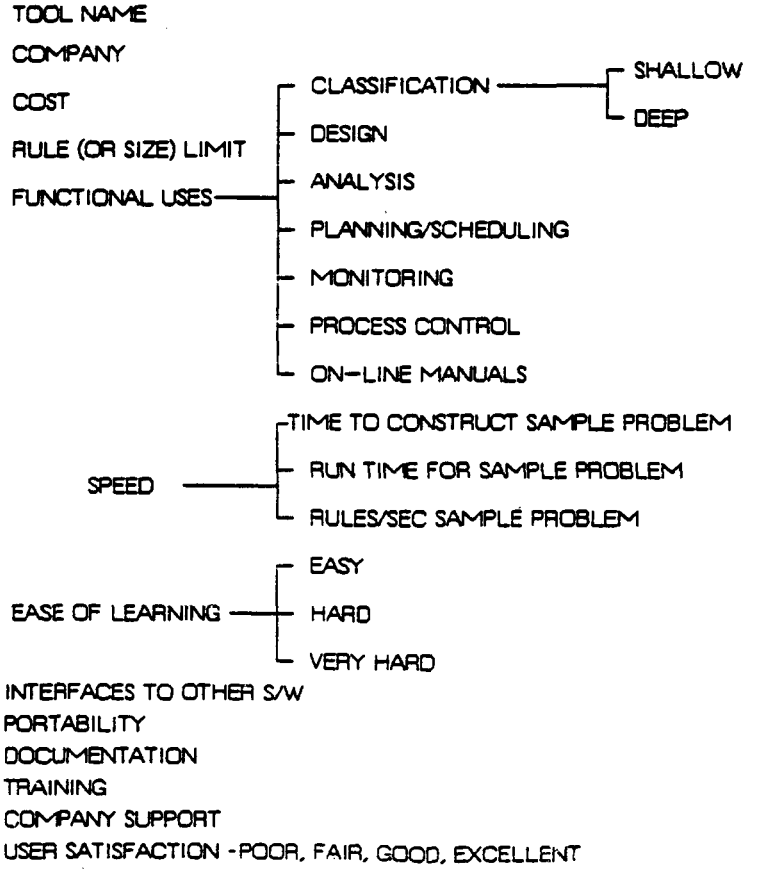

Figure 8.- Overall usability of tool.

APPENDIX A
BRIEF DESCRIPTIONS OF THE ESBTs COVERED IN THIS MEMORANDUM

ART

ART is a versatile tool incorporating a sophisticated programming workbench for use with advanced computers and work stations such as the Symbolics, LMI, and VAX. ART's strongpoint is "viewpoints," a technique that allows hypothetical nonmonotonic reasoning in which multiple solutions are carried along in parallel until constraints are violated or better solutions are found. At such points, inappropriate solutions are discarded. ART provides graphical interfaces for browsing both its viewpoint and "schema" (frame) networks. ART is primarily a forward chaining system with sophisticated user-defined pattern matching based on an enhanced version of an indexing scheme derived from OPS5 (discussed later). Object-oriented programming is made available via a procedural attachment (active value) feature to objects (schemata). ART has a flexible graphics workbench with which to create graphics interfaces and simulations. ART was designed for near-real-time performance, compiling its frame-based as well as relational knowledge into logic-like assertions (discrimination networks) to help achieve this performance. Applications particularly suited for ART are planning/scheduling, simulation, configuration generation, and design. Currently written in LISP, ART employs a very efficient unique memory management system (that virtually eliminates "garbage collection"). A C version is expected shortly.

KEE

KEE, which runs on advanced AI computers, is the most widely used programming environment for building sophisticated expert systems. An important aspect of KEE is its multifeature development and end-user interfaces incorporating windows, menus, and graphics. KEE contains a sophisticated frame system which allows the hierarchical modeling of objects with multiple forms of inheritance. KEE also offers a variety of reasoning and analysis methods, including object-oriented programming, forward and backward chaining of rules, hypothetical reasoning (KEE WORLDS), a predicate logic language, and demons. It has an open architecture which supports user-defined inference methods, inheritance roles, logical operators, functions and graphics. KEE has a large array of graphical interfaces under user control, including facilities for graphical simulation (SIM KIT, extra cost). KEE has been used for applications in diagnosis, monitoring, real-time process control, planning, design, and simulation.

Knowledge Craft

Knowledge Craft (KC) is a hybrid tool based upon frames with user-defined inheritance. It is an integration of Carnegie versions of OPS5, Prolog and the SRL frame representation language. It is a high productivity tool kit for experienced knowledge engineers and AI system builders. Frames are used for declarative knowledge; procedural knowledge being implemented by attached demons. KC is capable of hypothetical (nonmonotonic) reasoning using Contexts (alternative worlds). Search is user-defined. A graphics simulation package (Simulation Craft) is available. Designed to be a real-time system, KC is particularly appropriate for planning/scheduling and to an extent for process control, but it is somewhat of an overkill for simple

classification problems.

PICON

PICON is designed as an object-oriented expert system shell for developing real-time, on-line expert systems for industrial automation and other processes which are monitored using sensors in real-time, such as in some aerospace and financial applications. PICON operates on the LMI Lambda/PLUS LISP machine and the TI Explorer, which combine the intelligent processing power of a LISP processor with the high-speed numeric processing and data-acquisition capabilities of an MC68010 processor. The two processors operate simultaneously enabling PICON to monitor the system in real time, detect possibly significant events in process, diagnose problems, and decide on an appropriate course of action. PICON's icon editor and graphically oriented display enable a developer with minimal AI training to construct and represent a deep model of the process being automated. Rules about the process are entered using a menu-based natural language interface. PICON supports both forward and backward chaining.

S.1

S.1 is a powerful commercial ESBT aimed at structured classification problems. Facts are expressed in a frame representation; judgmental knowledge as rules. Though ostensibly a backward chaining system, it does forward reasoning using a patented "procedural control block" technique. Control blocks can be viewed as implementations of flow diagrams that procedurally guide the system as to the next steps to take in the current situation. Control blocks can invoke other control blocks, rules or an interactive dialogue. Control blocks are a powerful knowledge-based means of controlling the search and thus have made it possible to write programs containing thousands of rules without being overwhelmed by a combinatorial explosion (run times tend to be linear with the number of rules). S.1 is written in C and executes very rapidly. A major advantage of S.1 is that it can readily be integrated with existing software. A delivery version is available (without the development portion of S.1) and can be completely embedded in applications. S.1 is not aimed at exploratory programming, but is aimed at commercial applications where an iterative solution to solvable problems is desired. S.1 has an excellent user interface with graphical, mouseable representations of both the knowledge bases and the inference traces. Problems can be solved in terms of subproblems which can be linked (with consistency checking being performed) to handle the complete problem. All S.1 features are expressed in an integrated, strongly typed, block-structured language that facilitates system development and long-term maintenance.

ES ENVIRONMENT/VM or MVS (ESE/VM or ESE/MVS)

ESE is an improved version of EMYCIN designed for classification problems but does allow for forward chaining. It consists of two components: a development interface and a consultation interface. A Focus Control Block mechanism has been added to allow the developer to modify and control the flow of inference to increase the system speed. ESE/VM and ESE/MVS have good utilities to enable the developer to fashion the user interface and incorporate graphics when appropriate. ESE is particularly suitable for IBM mainframe users who must interface with existing software and databases.

ENVISAGE

ENVISAGE is a Prolog-derived tool. Thus, instead of entering rules, one enters logical assertions. Features beyond Prolog include demons, fuzzy logic, and Bayesian probabilities. ENVISAGE is primarily aimed at classification problems.

KES

KES is a three-paradigm system supporting production rules, "hypothesize and test" rules (using minimum set coverage to account for data), and Bayesian-type rules for domains in which knowledge can be represented probabilistically. KES is primarily geared to classification-type problems. KES can be imbedded in other systems. The hypothesize and test approach starts with a knowledge base of diagnostic conclusions (classifications) with their accompanying symptoms (characteristics). The session begins by the system selecting the set of all diagnoses that match the first symptom of the given problem and then reduces this set as the remaining problem symptoms are considered. If the initial set of diagnoses do not include all the remaining symptoms, new diagnoses are added to the set to cover these cases.

M.1

M.1 is a PC-based ESBT targeted for solvable problems rather than for exploratory programming It is a basically a backward chaining system designed for classification. It includes the capability for meta-level commands to direct forward reasoning. Written in C, it can readily be integrated with existing conventional software. Its main drawback is that it has no true object-description capability and therefore cannot readily support deep systems. However, M.1 does have a good set of development tools and friendly developer and user interfaces.

NEXPERT OBJECT

NEXPERT OBJECT is a powerful rule-based tool coded in C to run on the Macintosh with 512K of RAM, the Mac Plus, or the IBM PC AT. It has editing facilities comparable to those found on a large tool designed to run on the more sophisticated AI machines. The system allows the developer to group rules into "categories" so they need be called up only when appropriate. NEXPERT OBJECT supports variable rules and combinations of forward and backward chaining. The system can automatically generate graphical representations of networks of rules to indicate how they relate to each other. Similar networks can be generated to show rule firings in response to a particular consultation. NEXPERT OBJECT includes the capability for frame representations with multiple-inheritence and pattern matching rules so that deep reasoning is facilitated. NEXPERT OBJECT is a sophisticated system with a focus on graphical representation of the knowledge bases and reasoning process that enables a natural and comprehensible interface for both the developer and end-user.

PERSONAL CONSULTANT+ (PC+)

PC+ is an attempt to provide on a personal computer many of the advanced features found in more sophisticated tools such as KEE. Thus PC+ provides frames with inheritance as well as rules. PC+ supports the backward chaining derived from EMYCIN. It also includes forward chaining capabilities without variable bindings. PC+ has an extensive set of tools with user-friendly interfaces for both development and execution. The new 2.0 version supports up to 2 Mbytes of expanded or extended memory for increased knowledge base capacity. It also supports the IBM Enhanced Graphics Adapter and access to the popular dBase II and III data base packages on the PC. A version of PC+ is also available for the TI Explorer symbolic workstation. PC Easy, a simplified version of PC+ without frames, is also offered.

EXSYS 3.0

EXSYS 3.0 is written in C for PCs as an inexpensive, rule-based, backward chaining system oriented toward classification-type problems. Rules are of the if-then-else type. EXSYS includes a run-time module and a report generator. EXSYS can interface to California Intelligence's after-market products: FRAME to provide frame-based knowledge representation, and TABLET to provide a blackboard knowledge-sharing facility using tables.

EXPERT EDGE

EXPERT EDGE is basically a rule-based backward chaining system aimed at rapidly prototyping and delivering classification-type problems in the 50-to 500-rule range. It uses probabilities and Bayesian statistics to handle uncertainties and lack of complete information. Its outstanding feature is its excellent developer and end-user interface featuring a pop-up windowing environment. This is accompanied by a natural language interface and very good debugging facilities. The professional version interfaces with a video disk and is able to do extended mathematical calculations.

ESP ADVISOR and ESP FRAME-ENGINE

ESP is a Prolog-based system that is particularly appropriate for designing expert systems that guide an end-user in performing a detailed operation involving technical skill and knowledge. The developer builds the system by programming in KRL (Knowledge Representation Language); a sophisticated and versatile language supporting numeric and string variables including facts, numbers, categories, and phrases. PROLOG's heritage is clearly apparent in the system's ability to support a full set of logical operators, enabling the developer to write efficient complex rules. The ESP consultation shell offers a well-designed multipanel display that makes good use of color. A "text-animation" feature allows the developer to insert text at any point in a consultation. Though ESP Advisor has been designed as an introductory prototype tool, its extensibility makes more complex expert systems possible. The more sophisticatd ESP Frame-Engine supports frames with inheritance, forward and backward chaining rules, and demons.

INSIGHT 2+

INSIGHT 2+ is primarily a rule-based backward chaining (goal-driven) system but it can support forward chaining as well. Facts are represented as elementary objects with single-valued

or multivalued attributes. Rules are entered in PRL (Production Rule Language). The knowledge base is compiled prior to run time. Uncertainty is handled using "confidence factors" and thresholds. Because INSIGHT 2+ lacks methods for representing deep models, it is best used for heuristic problems, for which it is a serious tool. Its ability to access external programs and data bases is a major enhancement.

TIMM

TIMM is an inductive system that builds rules from examples. Examples are first translated into rules, which are then used to build more powerful generalized rules. TIMM handles contradictory examples by averaging the certainty of their conclusions. Partial-match analogical inferencing is used to deal with incomplete or nonmatching data. TIMM indicates the reliability of its results. The resultant expert systems can be embedded in other software programs.

RULEMASTER 3.0

Though RULEMASTER is capable (independently) of both forward and backward chaining, its major distinguishing feature is its inductive capability for generating rules from examples. It also offers fuzzy logic. The knowledge base interaction is accomplished via a text editor. If they prefer, knowledge engineers can develop RULEMASTER applications by writing code directly in the high-level Radial language of RULEMASTER instead of using examples. However, a strong programming background is required for facile usage. RULEMASTER can generate C or FORTRAN source code for fast execution, compactness, and portable expert systems that can interface to other computer programs.

KDS3

KDS3 inductively generates rules from examples. Examples can be grouped to develop knowledge modules (which KDS calls frames) which can be chained together to form very large systems. Both forward and backward chaining is supported. KDS3 can take input from external programs and sensors and can drive external programs. The resultant programs can be made interactive or fully automatic for intelligent process control. The entire system is written in assembly language for very rapid execution on PCs. Graphics can be incorporated automatically from picture files or drawn in real time using built-in KDS3 color-graphics primitives. KDS3 incorporates a blackboard through which knowledge modules can communicate. KDS2 without the blackboard facility is also available.

1st-CLASS

1st-CLASS is an induction system that generates decision trees (elaborate rules) from examples given in spreadsheet form. Problems can be broken into modules of examples, which can be forward or backward chained together. Rules can also be individually built or edited in graphical form on the screen. Several algorithms are available for inferencing: the system can match queries to examples that exist in the data base, or the system can utilize the rule trees as generated, or in the preferred mode which employs optimized rule trees which ask questions in the best order. Because all the rules are compiled, the system is very fast. 1st-CLASS is designed to readily interface with other software.

OPS5

Various versions of the OPS5 expert system development language, developed at Carnegie Mellon University, are available. OPS5 is a forward-chaining production rule tool in which many famous systems used at DEC, such as R1/XCON, have been built. OPS5 pattern-matching language permits variable bindings. However, OPS5 does not have facilities for sophisticated object representations. In general, the development environment is unsophisticated, although some debugging and tracing capability is usually provided. The use of a sophisticated indexing scheme (the Rete algorithm) for finding rules that match the current data base, makes OPS5 one of the fastest executing tools. Unfortunately, it is not an easy tool for the nonprogrammer to use. Representative versions, OPS5+, for the IBM PC, the Macintosh, and the Apollo Workstation, are available from Computer*Thought, Plano, TX.

APPENDIX B
ATTRIBUTES OF THE COMMERCIAL ESBTs

NAME	ART 3.0	KEE 3.0	KNOW-LEDGE CRAFT	PICON	S.1	KE ENVIRON/ VM OR MVS	ENVISAGE	KES	M.1	HESPERT OBJECT	PERSONAL CONSUL.+	EXSYS. 3.0	EXPERT EDGE	ESP FRAME ENGINE	INSIGHT 2+	TIMM	RULE-MASTER	KDS 3	1st CLASS
FUNCTIONAL USES																			
Classification Design	X X	X X	X X	X	X X	X	X	X	X	X X	X	X	X	X X	X	X	X X	X X	X
Planning/Scheduling Process Control	X X	X X	X X	X						X X						X	X	X X	
KB REPRESENTATION																			
Rules Structured Rules	X	X	X	X	X	X	X	X	X	Categories	X	X	X	X	X		X		X
Certainty Factors Rule Limit					X	X	X	X	X 1000	3000	999	X 5000	X		X 3000	X	X	X 16000 from 4000 examples	X
Frames w/Inherit. Object-Oriented	X X	X X	X X	X X	No inherit.					X	X			X				X	
Logic	X	X	X		X									X					X
Examples Structured Examples																X	X	X	X
Procedures	X	X	X		X	X	X	X		X	X			X					
INFERENCE ENGINE																			
Forward Chaining Backward Chaining	X (post rules)	X X	X X	X X	FR X	X X	X X	X	FR X	X X	X X	X X	X	X X	X X	X	X X	X X	X X
Demons Blackboard-Like	X X	X	X	X	X		X		X	X	X			X				X	X
Time-Modeling Truth Maintenance	X X	X X		X						X			X					X	
Meta Control Logic	X X	X X	X X	X	X X	X	X		X	X	X			X X					
Induction Math Calculations	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X
Context (Viewpoints, Worlds)	X	X	X																
PATTERN MATCHING																			
Variables	X	X	X	X				X	X	X		Numeric	Numeric	Numeric & String	Numeric		X		Numeric & Logical
Literals	X	X	X	X	X					X									
Sequences	X		X	X			X					X							
Segments	X		X	X				X					X						X
Wildcards	X		X																
DEVELOPER INTERFACE																			
KB Creation Word Processor Line Entry	X	X	X X	X	X X	X	X	X X	X	X	X	X X	X	X	X	X	X	X	X X
KB Editor Menu	X X	X X	X	X	X X	X			X X	X X	X X	X X	X		X	X	X	X	X X
Check for Consistency Graphics Rep of KB	X	X X	X	X X	X X				X	X X	X	X	X	X X	X	X	X	X	X X
Inference Tracing Graphics Utilities to build end-user interface	X X	X X	X X	X X	X e	X	X	X		X X	X	X	X	X	X	X		X X	X
Screen Format Utilities Graphics Simulation Capabilities	X X	X Sim Kit (Extra cost)	X Simulation-Craft (Extra cost)		X X	X			X	X X	X	X	X		X	X		X	X e
Why How	X X	X X	X X	X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X		X X	X X	TREE
Explanation Expansion Online Help	X X	X X	X X	X	X X	X X	X X	X	X X	X X	X X	X X	X X	X X	X	X	X X	X X	X X
Syntax Help Tool Extensible	X X	X X	X X	X X	X		X X		X	X X	X X		X X	X			X X	X X	X
SOFTWARE																			
Language of Tool Compilable	COMMON-LISP Incremental	COMMON-LISP Incremental	COMMON-LISP Incremental	ZETA-LISP,C X	C X	PASCAL Incremental	PASCAL X	C X	C X	C Incremental	LISP X	C Incremental	C X	Prolog 2 Incremental	Turbo-PASCAL X	FORTRAN X	C X	8086 Assembly	PASCAL X
Other Lang. Req. Operating Sys.	COMMON-LISP VMS, UNIX	COMMON-LISP	COMMON-LISP	LISP	UNIX VM/CMS VMS	CMS, CDOM, PASCAL/VS VM/CMS or MVS	VMS	MSDOS on PC	MSDOS	MSDOS VMS	MSDOS	MSDOS VMS	MSDOS	MSDOS	MSDOS		C Compiler DOS 2.0 UNIX, VMS	MSDOS	MSDOS
COMPUTERS SUPPORTED																			
Symbolics LMI	X X	X X	X X	X															
TI Explorer VAX	X X	X	X X	X	VMS ULTRIX		X	X				VMS UNIX				X	X		
XEROX IBM PC		X						X	X	AT	X	X	X	X	X	X	AT	X	X
Macintosh TI Pryd										X	X					X			X
APOLLO SUN	X	X X			X X			X X									X X		
IBM 370 Others		X	X		X X	X	X	X		X					X	X	X		
SYSTEMS INTERFACE																			
Lang. Hooks DB Hooks	Via Host Computer Via Host Computer	Via Host Computer Via Host Computer	Via Host Mach Via Host Mach	Via Host Mach Via Host Mach	C, Others Via *	X X	PASCAL, etc. X	C X	X X	C, PASCAL OBASE III	LISP OBASE III	X X	X X	Prolog 2 Via *	X OBASE II	X	X	PASCAL, BASIC Reads Assem. Lang.	ANY X
END USER INTERFACE																			
Screen Capabilities Line Menu	X	X X	X X	X X	X X	X X	X X	X X	X	X	X	X X	X	X X	X X	X	X	X	X
Graphics Simulation	X X	X Sim Kit	X Simulation-Craft	X X	e e			e		X X	e	e						X X	e
Why How	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X X	X	X X	X X	X
Help Initial Pruning	X X	X X	X	X	X	X	X	X	X	X	X X	X	X X	X	X X		X	X X	X
Multiple Solution Examples Saved	X	X	X X	X	X	X	X	X X		X X	X X	F X	X X	X		X		X X	X
Mult. & Uncertain User Resp. OK What If	X X	X X	X	X	X X	X	X	X X	X	X	X X	X X	X		X X	X		X X	X
COMPANY	Inference	Intellicorp	Carnegie Group	LMI	Tekno-ledge	IBM	Sys. Designers Software	S/W Arch. & Engr.	Tekno-ledge	Neuron Data	TI	Exsys, CA small	Human Edge	Exp Sys Inter	Level-5-R	GRC	Radian	KDS Gen Sys	Prop. In Motion
COST (1st unit)	$65K	$32K Includes training	$36K Includes training	$60K	$25K	$35K	$35K on VAX $15K on MicVAX	$4K PC $7K 80x86mx $25K VAX	$5K	$5K PC-AT $2K MAC	$3K	$395 PC $8K VAX	$2500 Adv $6000 Pro	$695	$495	$1.9K PC $15K others	$895 PC $6K 9Roll $17.5K VAX	$1495	$495
Run Time Sys. Cost	$1-6K	PC Host + VAX $20K + 6.5K/PC	None Yet		$9.5K	$25K		10% of Develop Sys.	~ $5K	$1K	$75 First Unit	One Time Fee	$5K	By Agreement	Nogot	No Fee	$100 PC $300 UNIX/VMS	Based on Quality	No Fee

NOTE e external programs; FR forward reasoning

APPENDIX C
TIMING BENCHMARK RESULTS (IN SECONDS) USING ESBTs TO SOLVE MONKEY AND BANANAS PROBLEM[1]

The Artificial Intelligence Section of the Mission Planning and Analysis Division at Johnson Space Center has made timing tests of several ESBTs. The benchmark used for testing was a modified version of a planning problem known as the monkey and bananas problem in which a monkey must plan how to overcome a series of obstacles in order to eat a bunch of bananas. The resulting benchmark for the forward chaining production systems consisted of 30 rules and required 81 to 86 rule firings to obtain the solution (depending on the production system used). Table C-1 presents the time in seconds required to reach the solution for various ESBTs.

While benchmarks provide useful information, one should avoid drawing far-reaching conclusions from them. This benchmark is just one of many possible benchmarks and does not fully test the capabilities of the tools.

TABLE C-1.– TIMING TESTS OF EXPERT SYSTEM BUILDING TOOLS

TOOL	INFER METHOD			COMPUTER						
	FC	BC	OO[a]	SYM	LMI	TI-EXP	VAX11/780	IBM AT	IBM PC	MAC
ART (V2.0)	X			1.2	3	2.4				
ART (V2.0)		X		7.6						
ART (V BETA 3)	X						17			
KEE[b] (V2.1.66)		X		17.8[c]						
KEE (V2.1.66)			X	0.4						
OPS5 (VAX V2.0)	X						1.3			
OPS5 (FORGY VPS2)	X			1.7						
OPS5+[d] (V2.0002)	X							5.2	19	14
EXPER-OPS5[e] (V1.04)	X									55
PROLOG		X		<0.25						
C-PROLOG		X					<0.5			

[a]OO REFERS TO OBJECT–ORIENTED

[b]KEE BENCHMARKS BASED UPON IMPLEMENTATIONS BY INTELLICORP. THESE BENCHMARKS ARE FOR INTELLICORP MODIFICATIONS OF FC KEE PROGRAMS WRITTEN BY THE JSC AI SECTION, WHICH TOOK ABOUT 10 TIMES AS LONG TO RUN. ALL OTHER BENCHMARKS EXCEPT PROLOG ARE BASED UPON IMPLEMENTATIONS BY THE JSC AI SECTION. PROLOG TIMING VALUES BASED UPON PRELIMINARY RESULTS AT NASA AMES

[c]TEST RESULTS USING A BETA-TEST BC RULE-COMPILER AT INTELLICORP YIELDED 0.3 sec FOR THE COMPILED VERSION OF KEE 3.0

[d]OPS5+ PRODUCT OF COMPUTER*THOUGHT, PLANO, TX

[e]EXPER-OPS5 PRODUCT OF EXPERTELLIGENCE, INC., SANTA BARBARA, CA

[1]This appendix extracted from Riley (1986).

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-88331, Rev. 1		

4. Title and Subtitle	5. Report Date
The Nature and Evaluation of Commercial Expert System Building Tools: Revision 1	March 1987
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
William B. Gevarter	A-87076
	10. Work Unit No.
	549-01-21

9. Performing Organization Name and Address	11. Contract or Grant No.
Ames Research Center Moffett Field, CA 94035	
	13. Type of Report and Period Covered
	Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, DC 20546	

15. Supplementary Notes

Point of Contact: William B. Gevarter, Ames Research Center, M/S 244-7
Moffett Field, CA 94035, (415) 694-6525 or FTS 464-6525

16. Abstract

This memorandum reviews the factors that constitute an Expert System Building Tool (ESBT) and evaluates current tools in terms of these factors. Evaluation of these tools is based on their structure and their alternative forms of knowledge representation, inference mechanisms and developer/end-user interfaces. Next, functional capabilities, such as diagnosis and design, are related to alternative forms of mechanization. The characteristics and capabilities of existing commercial tools are then reviewed in terms of these criteria.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Artificial intelligence, Expert systems, Expert systems building tools, Knowledge engineering, Knowledge representation, Inference engines	Unclassified - Unlimited Subject Category - 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	28	A02

For sale by the National Technical Information Service, Springfield, Virginia 22161